UNITED STATES PATENT APPLICATION

FOR

METHOD AND SYSTEM FOR DATA FLOW CONTROL OF EXECUTION NODES OF AN ADAPTIVE COMPUTING ENGINE (ACE)

Inventor(s):

W. James Scheuermann

Joseph A. Sawyer, Jr.
Sawyer Law Group LLP
2465 E. Bayshore Road, Suite 406
Palo Alto, California 94303

METHOD AND SYSTEM FOR DATA FLOW CONTROL OF EXECUTION NODES OF AN ADAPTIVE COMPUTING ENGINE (ACE)

FIELD OF THE INVENTION

The present invention relates to data flow control during task execution in an adaptive processing system.

BACKGROUND OF THE INVENTION

The electronics industry has become increasingly driven to meet the demands of high-volume consumer applications, which comprise a majority of the embedded systems market. Embedded systems face challenges in producing performance with minimal delay, minimal power consumption, and at minimal cost. As the numbers and types of consumer applications where embedded systems are employed increases, these challenges become even more pressing. Examples of consumer applications where embedded systems are employed include handheld devices, such as cell phones, personal digital assistants (PDAs), global positioning system (GPS) receivers, digital cameras, etc. By their nature, these devices are required to be small, low-power, light-weight, and feature-rich.

In the challenge of providing feature-rich performance, the ability to produce efficient utilization of the hardware resources available in the devices becomes paramount. As in most every processing environment that employs multiple processing elements, whether these elements take the form of processors, memory, register files, etc., of particular concern is controlling the flow of data and task execution within and among the multiple processing elements. Accordingly, what is needed is a manner of controlling the flow of

data for efficient task execution in an adaptive processing system. The present invention addresses such a need.

SUMMARY OF THE INVENTION

Aspects for data flow control of execution nodes of an adaptive computing engine (ACE) are presented. The aspects include associating task parameters with tasks within an execution node. Readiness of task resources is identified based on a status of the task parameters. Subsequently, allocation of the tasks to the execution node occurs based on the readiness of task resources.

Through the present invention, data flow control techniques are achieved that provide for efficient and straightforward task execution pacing in execution nodes of an adaptive processing system. These and other advantages will become readily apparent from the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates an execution node diagram of an adaptive computing engine (ACE) in accordance with the present invention.

Figure 2 illustrates a more detailed illustration of the execution node.

Figure 3 illustrates a block flow diagram of data flow control for task execution within the execution node in accordance with the present invention.

Figure 4 illustrates a block diagram of a system for data flow control in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to data flow control during task execution in an adaptive processing system. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The aspects of the present invention relate to task flow control in an adaptive computing engine (ACE). In general, an ACE refers to a silicon die which integrates a number of compatible, high-performance, scalable computing elements, memories, input/output ports, and associated infrastructure to provide efficient implementations for various applications. As shown in Figure 1, the ACE principally include heterogeneous nodes 1000, a homogeneous network 1002 with a flexible input/output subsystem and bulk memory interfaces to high performance memory controllers and associated memories 1004, a system bus interface to a system processor (if present) 1006, system memory 1008, and their associated services, and an infrastructure that includes clocking 1010, configurable I/O (input/output) 1012, and power management, security management, and interrupts 1014. The heterogeneous nodes 1000 are interconnected suitably by the homogeneous network 1002 using a by-four fractal to support scaling to any number of nodes. A more detailed discussion on the node structure and interconnection network of the ACE is presented in co-pending U.S. patent application, serial no. 09/898,350, entitled "Method and System for an Interconnection Network to Support Communications Among a Plurality of Heterogeneous

Processing Elements", and filed July 3, 2001, which is assigned to the assignee of the present invention and incorporated herein by reference in its entirety.

A diagram for a node shown in Figure 2 illustrates that each node 1000 includes an execution unit 1020 coupled to memory 1006 in the form of registers 1022, network memories 1024, and data memories 1026, and network interfaces, network in/network out. Streaming the data within and among the nodes 1000 in the network 1002 for task execution is achieved through the aspects of data flow control in accordance with the present invention. The aspects are described with reference to nodes 1000 which contain one or more finite state machine-based, parameterizable, reconfigurable computational elements, i.e., a reconfigurable node (R-node), that capably performs any of a number of tasks, including, for example, signal processing, such as FIR filtering, IIR filtering, FFT, DCT, IDCT, convolution/correlation, convolutional encoding and Viterbi decoding, Huffman encoding and decoding, encryption and decryption, and LSFR/PN sequence generation. It should be appreciated that these aspects may be applied to other types of nodes, including, for example, fully programmable nodes, and hybrid nodes, which contain both programmable and reconfigurable execution units. In accordance with the present invention, data flow techniques are used to pace the task execution by the execution unit within the nodes, as presented with reference to the block flow diagram of Figure 3 and system diagram of Figure 4.

In considering an R-node having number 'k' input ports, 'k' output ports, 'm' fsm-controlled execution units (fsm), and up to 'i' instances for each fsm, the data flow control techniques of the present invention begin with construction of an active task list in the form of a queue (step 1100). In order to construct the task list, the status of task parameters is

determined (step 1110). These task parameters indicate whether a task is executable and ready for placement in the task list. For a given task to be executable, the necessary input buffer(s) and output buffer(s) must be available and the fsm(s) must be in an idle state. Once all the parameters have satisfied the condition requirements, the task is placed in the queue (step 1112).

The issuance of tasks from the queue proceeds based on the status of the fsms. Whenever all fsms are idle and the queue is not empty, the next executable task is read from the queue and the 'go' signal of the corresponding fsm is asserted (step 1114). If the current instance of the fsm is different form the previous instance, as determined via step 1116, the fsm is reconfigured (step 1118). Task execution ensues, i.e., data is read from the input port, processed, and written to the output port (step 1120). When the task completes, the 'done' signal is generated, and the fsm re-enters its 'idle' state (step 1122). The process continues by issuing a next executable task from the task queue.

Referring now to Figure 4, in determining the status of the task parameters, up/down counter flags suitably indicate availability of each input port/buffer 1200 and each output port/buffer 1202. A status of an idle signal for each fsm 1204 is also tracked 1206. As is further shown in Figure 4, a counter value 1208 is utilized as a signal for selectors 1210 receiving the flag status for a particular input port and output port and as a signal for a lookup table 1212 for selecting a corresponding fsm 1204 via a selector 1214. The signals from the selectors 1210 are combined logically, e.g. via an AND gate 1216, to provide a write signal to allow the task and its parameters to be added to the task list in the queue 1218. A data structure 1220 binds parameters identifying, by number, an input port, an output port, an fsm, and an instance of that fsm associated with each task, where an instance

refers to a variation in performance by a particular fsm. For example, an fsm may be configured to perform Viterbi decoding with different constraint lengths. Thus, for each constraint length, a separate instance of the decoder would be utilized. Decoder 1222 and selectors 1224 and 1226 are also included as part of the flow control logic for tracking fsm status during task execution.

With the present invention, data flow control techniques are achieved that provide for efficient and straightforward task execution pacing in execution nodes of an adaptive processing system. The techniques further provide for consistency and uniformity for application across any and all node types within the system. Thus, the techniques are well-suited to accommodate expansion in the network of nodes.

From the foregoing, it will be observed that numerous variations and modifications may be effected without departing from the spirit and scope of the novel concept of the invention. Further, it is to be understood that no limitation with respect to the specific methods and apparatus illustrated herein is intended or should be inferred. It is, of course, intended to cover by the appended claims all such modifications as fall within the scope of the claims.